Application and comparison of Machine Learning Algorithms for steel quality prediction

Diego Esteban Zapata Sanchez, Matrikel nummer - m12317103.

***Abstract* - The following report aims to present the application and comparison of machine learning algorithms for steel quality prediction in continuous casting plants with data from the 'Stahl- und Walzwerk Marienhütte GmbH Graz'. Therefore, given a dataset of sensor and process data, different models will be designed based on Machine Learning Algorithms to predict a quality relevant metric.**

Keywords: Quality, dataset, mode, machine learning algorithms.

I. OBJECTIVES

General objective:

Predict a quality relevant metric (yield strength) through the use of different machine learning algorithms, training the models with the given dataset of sensor and process data.

Specific objectives:

- Implement effective data set cleaning techniques and save the processed data.
- Make and sabe graphs to visualize the distribution and trends of the data.
- Perform analysis of the data set based on its statistics and graphs.
- Develop machine learning models to predict steel quality from the continuous casting plants dataset provided.
- Train the model to learn and map the input process data to the target steel quality variables.
- Compare various ML architectures to identify the most effective model for accurate steel quality prediction.
- Assess the performance of implemented models using appropriate evaluation metrics.

II. INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) have evolved significantly in recent decades due to advances in algorithms, massive data availability and computational power. ML, an essential branch of AI, has proven to be fundamental in allowing machines to learn patterns and make decisions without human intervention, transforming the way we approach problems and optimize processes in all types of industries, areas and environments, like from health to production.

Its importance lies in the ability to automate complex tasks, perform predictive analysis and offer efficient solutions, promoting innovation and improving decision making in real time. Thus, within the framework of the application of machine and deep learning techniques, this project focuses on the creation and optimization of predictive models to predict the quality of steel in continuous casting plants, based on a data set with 21 sensor and process data as influential variables in the creation of steel and its final quality result.

During the development of the project, various data preprocessing techniques were applied, such as separating the original data set into subsets with different functions, cleaning null values and their management to avoid losing information, and removing outliers that may influence the training of the model, in order to guarantee the quality and consistency of the information. Subsequently, Cross Validation and supervised learning algorithms were used to build predictive models, while the selection of hyperparameters was carried out using the Grid Search technique.

This project seeks not only to provide efficient solutions for predicting the quality of the steel produced, but also to lay the foundations for the application of similar methodologies in industrial and production contexts, promoting the effective implementation of machine learning techniques for the continuous improvement of processes and informed decision making, thus contributing to the improvement of efficiency and quality in industrial processes.

III. LITERATURE REVIEW

1. Access and preprocess the data

Data quality has a direct impact on the effectiveness and accuracy of the models, since they contribute to the creation of more accurate and generalizable models, which is why eliminating noise and errors in the data helps to avoid bias and improve quality of the resulting model.

Likewise, data sets often have missing values, either due to measurement errors or simply because certain data are not available. Therefore, methods such as imputation of null values or filling them with the mean, median or other appropriate value depending on the situation, can help manage these missing values to maintain the quality and efficiency of the final model [1].

On the other hand, ensuring that all variables have the same scale is crucial for algorithms that rely on distance measures, such as linear regression and support vector machines. In the case of this project, it is very important considering that classification is not done but regression, that is, predicting a value (steel quality/yield strength).

Finally, the presence of outliers can negatively affect some models. Identifying and handling outliers appropriately is essential to improve the robustness of the model. In addition, in data sets with many variables, dimensionality reduction can help improve model efficiency and reduce the risk of overfitting.

2. Exploratory data analysis through visualization

Visualizing statistics and data representation provides a complete overview of the distribution, relationships and patterns of the data, making it easy to identify trends and possible outliers, thus being a critical aspect of understanding and deriving insights from sets. of data.

Measures of central tendency (the mean, median, and mode) provide information about the central values around which the data are distributed. These measurements provide an idea of typical or representative values within the data set. Similarly, measures of dispersion (the standard deviation and interquartile range (IQR)) convey the dispersion or variability of the data. Understanding dispersion is crucial to assessing the degree of variability and potential outliers [2].

$$(Mean)\ \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad (SD)\ S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Likewise, exploring correlations between variables using coefficients such as Pearson's correlation helps identify linear relationships. Positive or negative correlations impact how changes in one variable relate to changes in another.

On the other hand, regarding visualizing the information in graphs, in the case of showing the distribution of a single variable, histograms show the frequency of different values [3], revealing patterns and possible asymmetries, while box plots provide a visual summary of distribution, central tendency and dispersion of the distribution and identify possible outliers [4].

Likewise, scatter plots are useful for examining relationships between two variables and revealing patterns such as trends, clusters, or outliers [5].

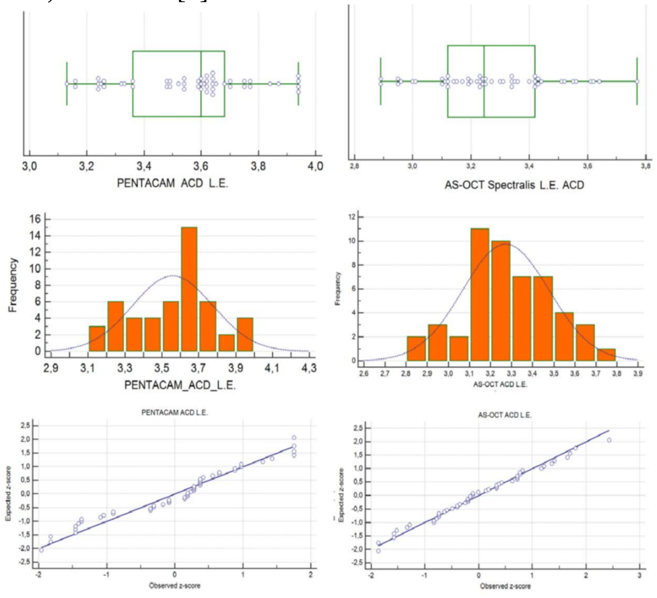

Figure 1: *Boxplots, histograms and scatter plots. Taken from https://www.researchgate.net/publication/343809489/figure/fig1/AS:927239248035841@1598082472354/Box-and-Whisker-plot-histogram-with-normal-curve-correlation-scatter-diagram-with.png*

Finally, heat maps are particularly beneficial for exploring correlations between multiple variables simultaneously, since they use color gradients to represent the strength and direction of relationships.

For all this, the visualization of data statistics and relationships improves the interpretability of complex data sets, which helps in decision making and guides subsequent steps in the process of data analysis and creation of machine learning models.

3. Cross Validation and Hyperparameters

In Machine Learning, effective model evaluation and tuning are pivotal for building robust and high-performing models. Cross-validation and hyperparameter tuning are two essential concepts in this regard.

Cross-validation is a validation technique used to assess a model's performance by splitting the dataset into multiple subsets. The model is trained on a portion of the data and validated on the remaining segments. This process is repeated multiple times, and the average performance metrics provide a more reliable estimate of the model's generalization capabilities. Common methods include k-fold cross-validation, where the data is divided into 'k' folds, and leave-one-out cross-validation (LOOCV), where each data point serves as a validation set.

Hyperparameters are external configurations that influence a model's learning process but are not learned from the data. They are set before training and play a crucial role in determining the model's performance. Examples include the learning rate, regularization strength, and the number of hidden layers in a neural network. Fine-tuning hyperparameters is essential to optimize model performance and prevent overfitting or underfitting.

Grid search and random search are common techniques for hyperparameter tuning. Grid search also applies Cross Validation and systematically evaluates combinations of hyperparameter values from predefined ranges, while random search explores random combinations. These methods assist in finding the optimal set of hyperparameters that maximizes the model's predictive power.

4. Machine Learning Algorithms (Model overview)

In predictive modeling, choosing the right algorithm is critical for achieving accurate and reliable predictions. Here it is provided an overview of four popular regression algorithms: Linear Regression, Random Forest, Support Vector Machine (SVM), and Neural Network (NN).

- **Linear Regression:** It is a fundamental algorithm used for predicting a continuous target variable based on linear relationships with input features. It assumes a linear connection between the features and the output. This model is suitable for cases where the relationship between variables can be adequately represented by a straight line.

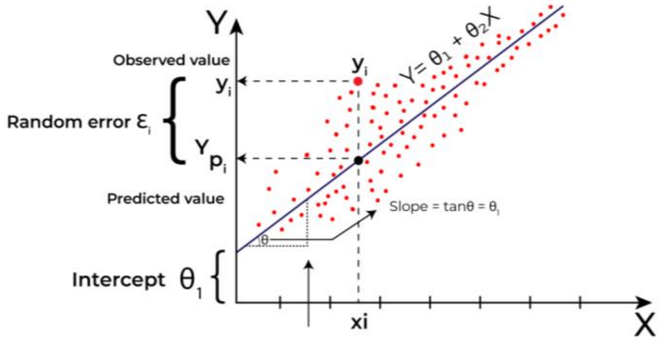
Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
Intercept θ_1
$Y = \theta_1 + \theta_2 X$
Slope = tanθ = θ_i
θ
xi
X

Figure 2: Linear Regression. Taken from https://media.geeksforgeeks.org/wp-content/uploads/20231129130431/11111111.png

- **Random Forest:** It is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the average prediction of the individual trees for regression tasks. This algorithm is known for its robustness, ability to handle complex relationships and resistance to overfitting.

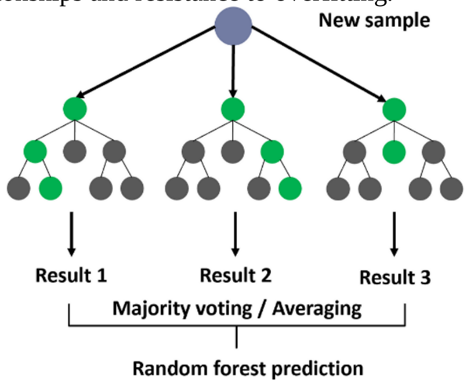

*Figure 3: Random Forest. Taken from https://miro.medium.com/v2/resize:fit:1400/1*jE1Cb1Dc_p9WEOPMkC95WQ.png*

- **Support Vector Machine:** It is a versatile algorithm capable of performing both classification and regression tasks. In regression, SVM aims to find a hyperplane that best represents the data while minimizing errors. It is effective in capturing complex relationships and is less sensitive to outliers compared to some other models [6].

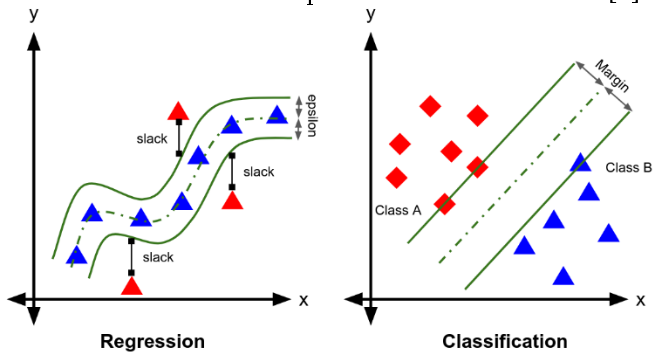

*Figure 4: Support Vector Machine. Taken from https://miro.medium.com/v2/resize:fit:1100/1*XE9jt0r1yAW8LnliQ3mllQ.png*

- **Neural Network:** Inspired by the structure of the human brain, consist of interconnected nodes arranged in layers. In regression tasks, neural networks can learn complex patterns and relationships in the data. They are particularly effective for capturing nonlinear mappings but may require careful tuning of hyperparameters.

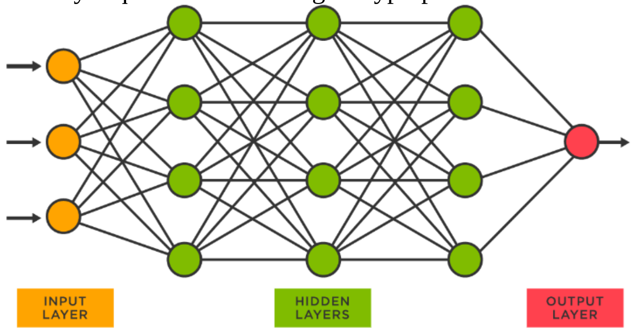

Figure 5: Neural Network. Taken from https://vitalflux.com/wp-content/uploads/2023/02/Sklearn-Neural-Network-MLPRegressor-Regression-Model-.png

IV. METHODOLOGY

Access and prepare (preprocess and cleaning) the data for Machine Learning tasks:

Data is loaded from CSV files corresponding to normalized training and test sets. The data is then combined and separated in training, testing and validation sets, applying preprocessing functions, such as shuffle_data and separate_and_clean_data, transforming all values into numeric (or null if it cannot be numeric) and dropping any row with null outputs. In addition, the resulting total data is saved in a new CSV file named "normalized_total_data.csv".de humedad.

- Sizes of the Total data set: X = (10979, 21), y = (10979,).

Later, the separate_and_clean_data function performs two main tasks. First, if the data is not pre-cleaned, it converts all values in the data set to the numeric type, assigning null values to those that cannot be converted. Then, it removes rows that have null values in the 'output' column and removes duplicates from the data set. Second, the function separates the data set into inputs (X) and outputs (y), and returns these two parts. Additionally, it prints the sizes of the resulting arrays, providing information about the structure of the clean, separated data. This process is essential to prepare data before using it in training machine and deep learning models [7].

- Sizes of the Training set: X = (8234, 21), y = (8234,).
- Sizes of the Testing set: X = (1646, 21), y = (1646,).
- Sizes of the Validation set: X = (1099, 21), y = (1099,).
- Sizes of the Total data set: X = (10979, 21), y = (10979,).

Then, the fill_null_values function is responsible for handling null values in the data set. First, it concatenates the inputs (X) and outputs (y) into a single DataFrame (df). Next, check if a set of means is provided. If not provided, calculates the means of the input columns (X). Next, replace the null values in X with the calculated means. Only with the testing set a set of means is provided, corresponding to the means of the training set so that the first one is effectively related and based on the second one, while this is not done with the validation set so as not to influence the next validation that will be done of the effectiveness of the trained model.

The function prints descriptive statistics of the data set, highlighting the total amount of data in all columns and whether null values are present. It also shows the means of the 21 inputs in tabular form.

The final step for processing data is to remove outliers. The remove_outliers function identifies and removes outliers in the data set. First, it concatenates the outputs (y) and inputs (X) into a single DataFrame (df). It then iterates over the input columns and uses the interquartile range (IQR) method to determine the upper and lower bounds, based on a default threshold = 3.75. This is done 5 times to ensure removal of outliers that are outside the set range each time it is updated. Values that fall outside these limits are considered outliers and are removed. The function prints the number of rows removed and the percentage of data removed from the original set. Importantly, the function does not remove outliers in the validation set (X_validation, y_validation) in order not to influence the next validation that will be done of the effectiveness of the trained model. This process helps improve the robustness of the model by removing data that could introduce noise or bias during training.

- 806 rows have been removed from Training set, which is the 9.788681078455186% percent of the original data.
- 116 rows have been removed from Testing set, which is the 7.047387606318348% percent of the original data.
- 1093 rows have been removed from Total data set, which is the 9.955369341470078% percent of the original data.

Finally, the save_data function takes the input data set (X and y) and saves them to CSV files with the specified name. If the data is not instances of pd.Series or pd.DataFrame, it converts it to those formats. Then, concatenate (y) and (X) into a single DataFrame (combined_df), sort the columns and rename the input columns as "input1", "input2", ..., "input21", and the output column as "output". Finally, save the DataFrame to a CSV file at the specified path (data_path) with the given file name (name_file). The function prints a message indicating that the file has been saved successfully, or displays an error message if a problem occurs during the process. In this case, the functions are used to save different processed data sets (training, testing, validation, and total) to specific CSV files.

Generate statistics to get insights of the data:

Data is loaded from CSV files corresponding to the total raw and processed data sets. The training, testing and validation data sets are not loaded, but rather the total raw and processed data to be able to make analyzes and comparisons between them through graphs and statistics.

First, the statistics corresponding to each of the columns, output and inputs, of the data sets are displayed and saved. It can be seen the total amount of data in the column, its mean, its standard deviation, its minimum value, its maximum value and its quartiles (25%, 50% and 75%). For example, these are the results for the output:

0. 0.490648, 0.092605, 0, 0.440000, 0.502222, 0.553299, 1

When viewing the results, it is evident that the data has already been scaled with minimum and maximum values between 0 and 1, and its mean is close to being in the middle of this range (0.49). However, no standardization was applied, since the mean is not 0 and the standard deviation is not 1.

The first tool used to visualize the data is box plots, then scatter plots, then heat maps and finally histograms:

1) Box plots:

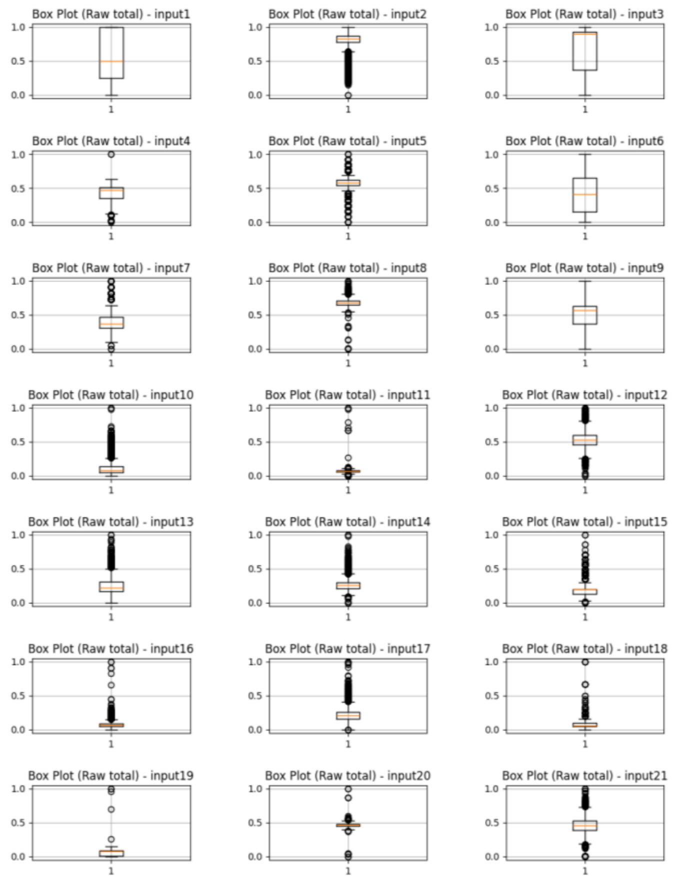

Figure 6: *Box plots for the raw total data set.*

In Figure 6 too many outliers are displayed in most of the 21 entries in the raw data set. The only ones that do not present atypical values are inputs1, 3, 6 and 9.

Most entries significantly improved the outlier problem they had in Figure 7. However, they continue to present atypical values that fall outside the interquartile range. Nevertheless, it was decided to establish a limit for removing outliers of 10%, and currently 1093 rows of data have already been removed, corresponding to 9.955% of the total database. Therefore, the removal of outliers was not continued.

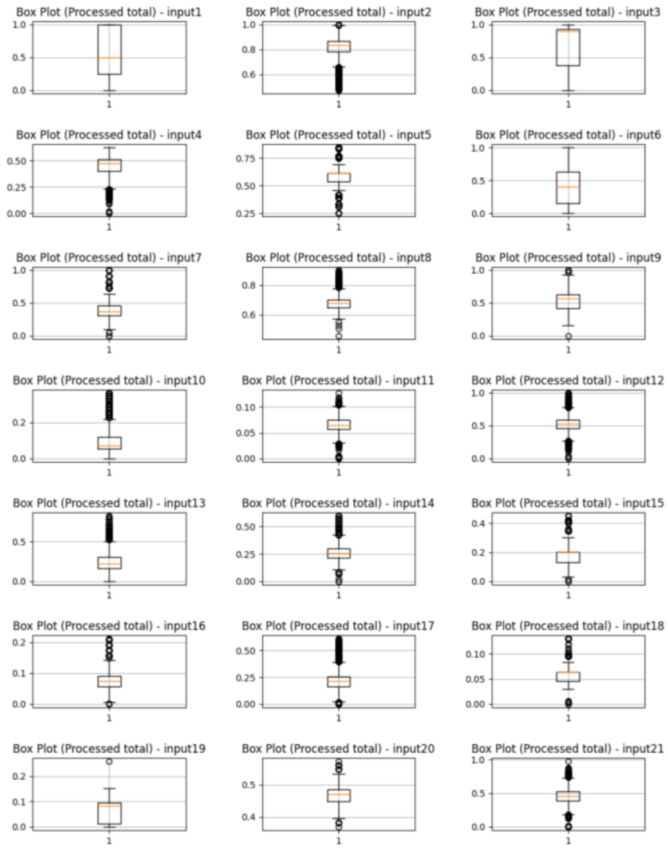

Figure 7: *Box plots for the processed total data set.*

2) Scatter plots:

It is seen in Figure 8 the distribution of the data corresponding to the 21 inputs. In inputs with lower standard deviation, such as number 19, the outliers are quite evident, while in others such as number 3, the outliers are not very clear and the interquartile range does not apply, since the data is distributed in 2 groups. Therefore, continuing to remove outliers would have eliminate the bottom grouping of data visible in the graph.

Then in Figure 9, the process of eliminating outliers allowed most of the data to be preserved and, at the same time, eliminate outliers that could influence the training of models with different Machine Learning algorothms.

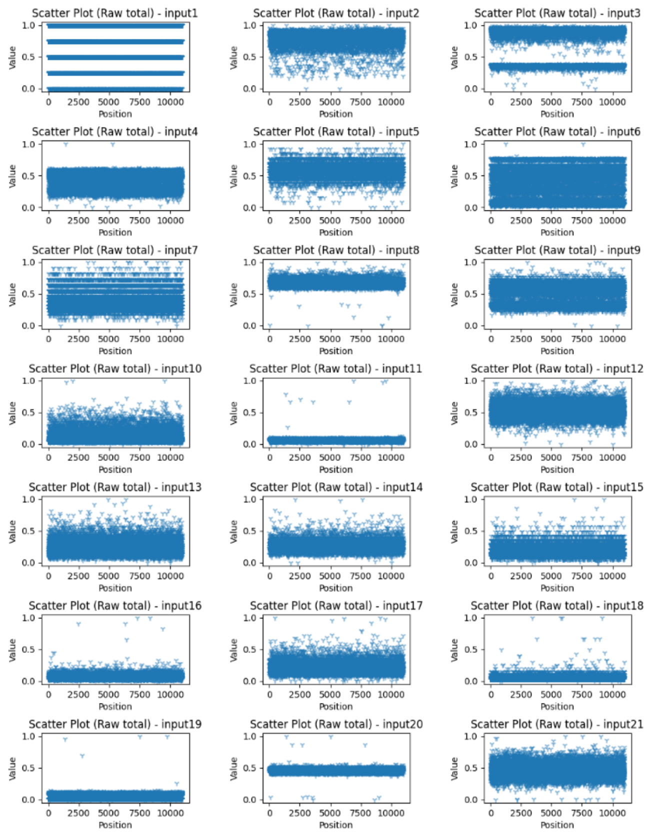

Figure 8: *Scatter plots for the raw total data set (just inputs).*

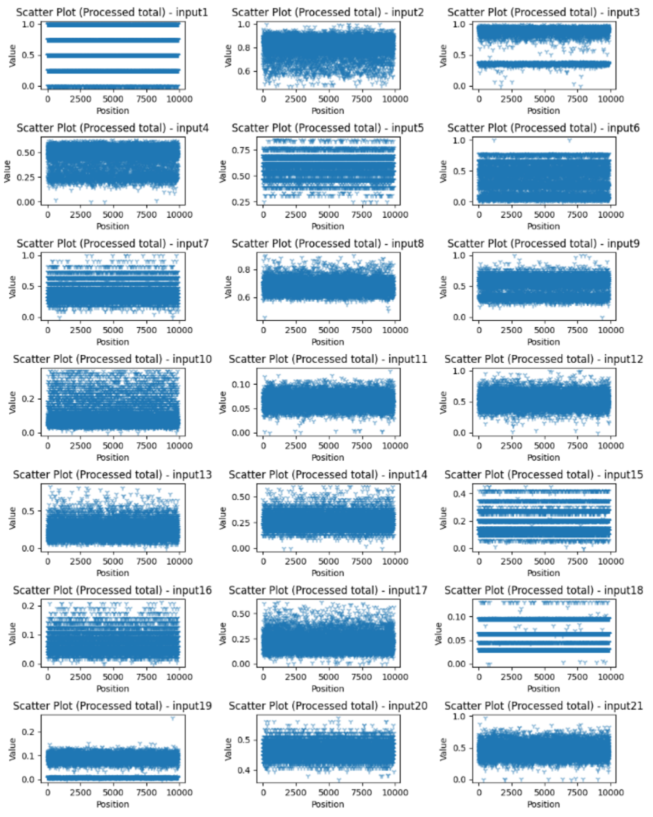

Figure 9: *Scatter plots for the processed total data set (just inputs).*

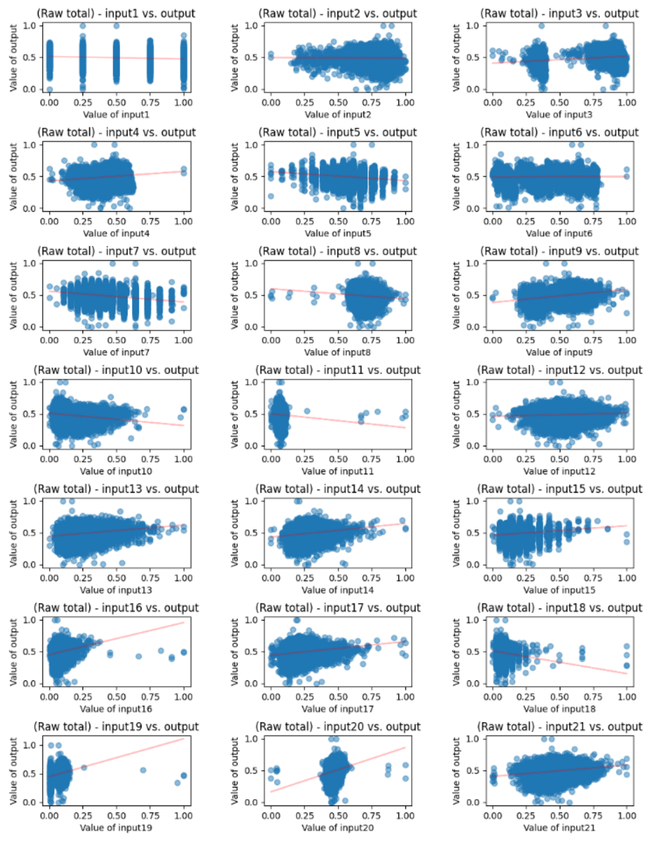

Figure 10: *Scatter plots for the raw total data set (inputs and output).*

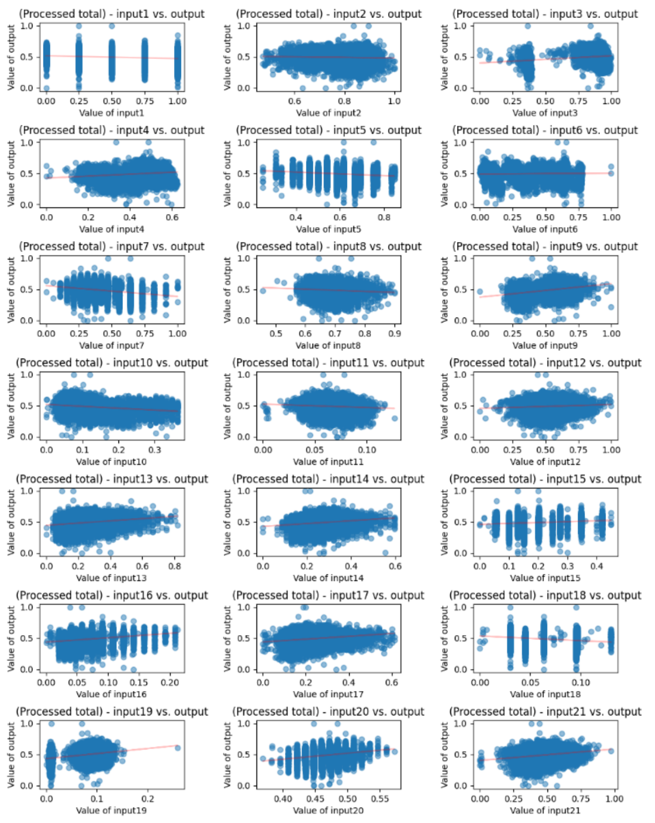

Figure 11: *Scatter plots for the processed total data set (inputs and output).*

It can be seen in Figure 10 the relationship between the value of each input (on the x-axis) with the output (the y-axis). While some inputs do not seem to show a great relationship with the output, since the slope of the relationship (the red line) is close to 0 (as in the case of input 1), others seem to have a very defined relationship. However, this may change, because the graphs displayed here are before data preprocessing.

Indeed, after the elimination of outliers, all relationships between inputs and output changed a bit, what can be seen in Figure 11.

3) Heat maps:

Thanks to the heat map in Figure 12, it is clear that the output does not present strong relationships with any of the inputs, while some inputs present strong relationships with each other.

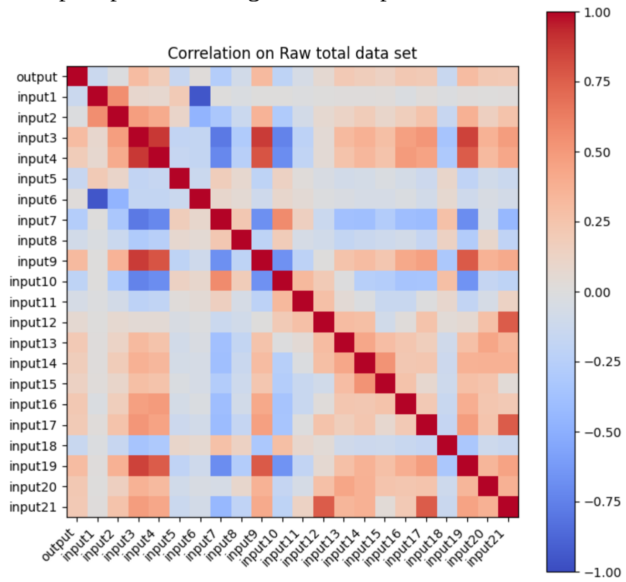

Figure 12: *Heat maps of correlation for the raw total data set.*

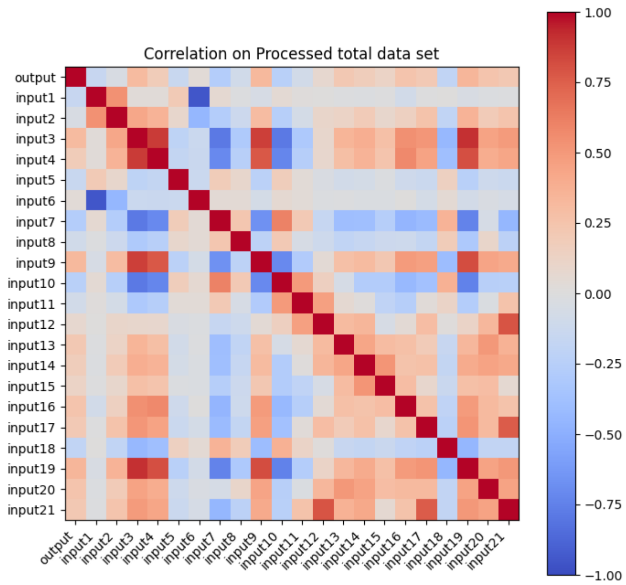

Figure 13: *Heat maps of correlation for the processed total data set.*

From what can be seen in Figure 13, the relationship between variables did not change much after data preprocessing. However, it is still useful to avoid problems with the training of machine learning models and, in addition, the heat map can also serve as a basis for making decisions related to the selection of features, as in the case of inputs 1 and 6, which are the ones that seem to have the least relationship with the majority of variables.

4) *Histograms:*

In the graphs of Figure 14 it can be seen the distribution of values that each input and output have, being located between half of the values and having a distribution similar to the normal one.

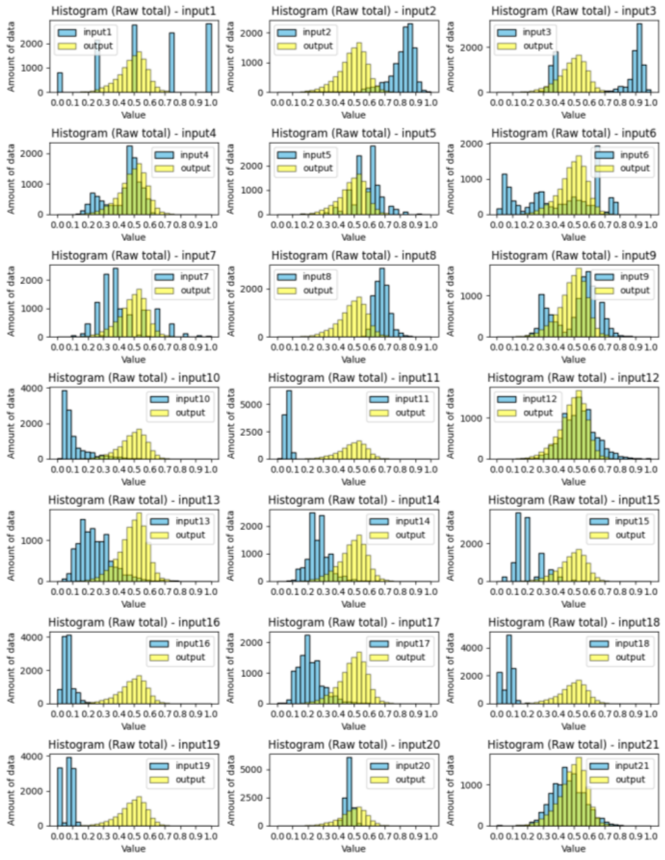

Figure 14: *Histograms for the raw total data set (inputs and output).*

The histograms of the processed data in Figure 15 are quite similar to those of the unprocessed data, since in reality the raw data for this project had already been cleaned and processed to some extent (scaled with minimum and maximum values between 0 and 1 and without null values). Although the raw data still had outliers and the processed one does not have, they are difficult to visualize on a histogram, since that is not the function of this type of graph.

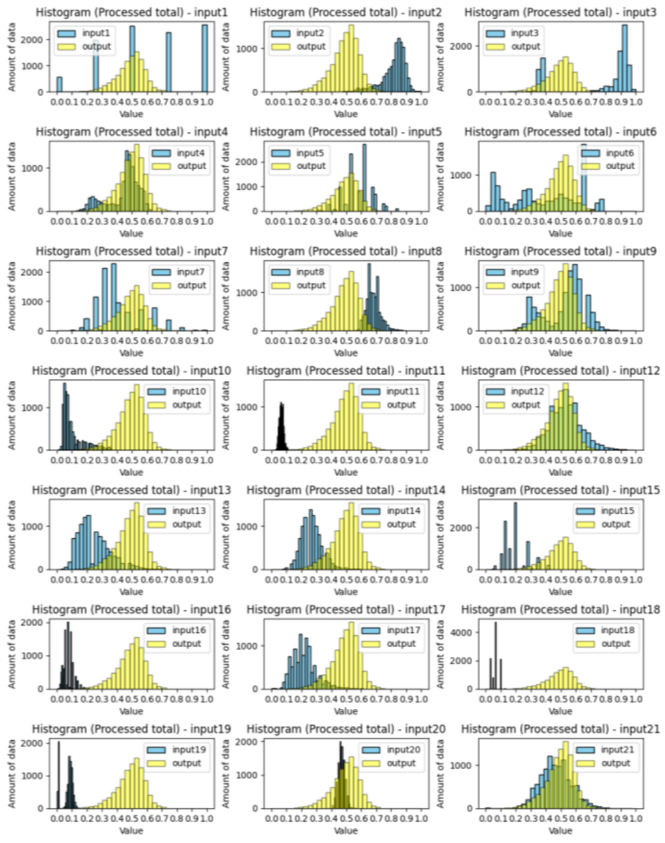

Figure 15: *Histograms for the processed total data set (inputs and output).*

Model Architecture:

Data is loaded from CSV files corresponding to the training, testing and validation data sets and then creates an empty DataFrame (results_df) with columns for various metrics and information related to model evaluation. Later, rows will be append to it as it is going through different models and collect their evaluation results.

4 machine learning models will be trained using distinct algorithms: Multiple Regression, Random Forest, Support Vector Machine and Neural Network, each with their respective hyperparameters.

The selection of the best-tuned model for each algorithm will be conducted through Cross-Validation using GridSearch. GridSearch explores various combinations of hyperparameters to identify the model with the lowest Mean Squared Error. A 10-fold Cross-Validation will be employed for all algorithms, providing mean and standard deviation results for the best model. Subsequently, the efficiency of the best model from each algorithm will be evaluated using the testing and validation datasets, assessing its performance based on Mean Squared Error. Furthermore, a learning curve will be plotted to analyze the training progress of each model.

a) *Multiple Regression:*

Parameters for training:
- {}.

Best parameters selected with the GridSearch:
- {}.

Mean and standard deviation of the results of the best model:
- -0.006 and (+/-0.001).

Mean Squared Error on testing dataset:
- 0.006287932364769719.

Mean Squared Error on validation dataset:
- 0.006485017062016511.

On the other hand, the learning curve reveals that initially, while the Mean Squared Error (MSE) of the training set decreases rapidly, the testing set exhibits a slower pace of improvement and tends to stabilize early. What's more concerning is that, after reaching its minimum, the error begins to rise with an increasing amount of data. Subsequently, it stabilizes concurrently with the testing set, and eventually at the end, both errors start to increase.

Unfortunately, if a model initially demonstrates improvement but later deteriorates as more information is introduced, it may be overfitting the training set. Initially, the model might have learned specific patterns from the training set, adapting to its particularities. However, as it encounters more data, those patterns and particularities may fail to generalize effectively to new and unseen data.

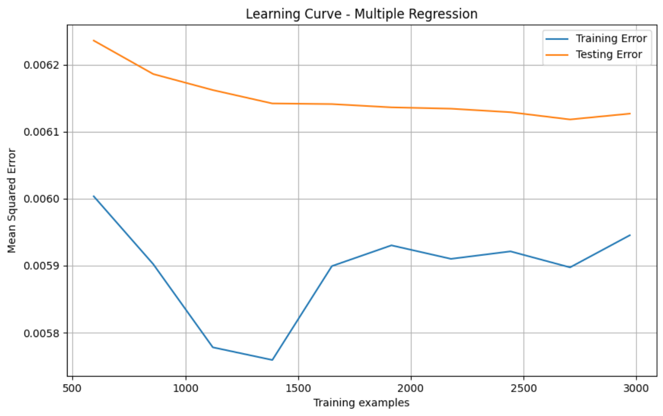

Figure 16: *Learning curve – Multiple Regression.*

In addition, the level of importance that each characteristic or input has for decision making or for reaching the expected output was also obtained. This can be very useful in terms of process optimization depending on the meaning or what each variable corresponds to.

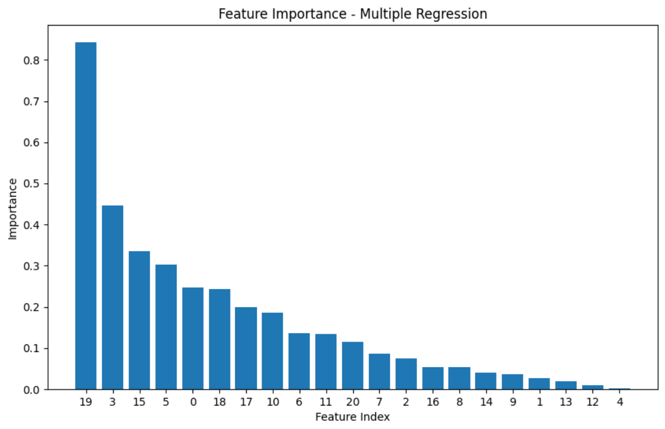

Figure 17: *Feature importance – Multiple Regression.*

b) *Random Forest:*

Parameters for training:
- {'n_estimators': [50, 125, 250, 500], 'max_depth': [None, 10, 25, 50]}

The parameter max_depth controls the maximum depth of individual trees in the forest. A higher value allows the trees to be deeper, which could result in a more complex model capable of capturing more detailed patterns in the training data. However, a very high value can lead to overfitting, especially if the data is noisy or if there are few examples in the training set.

The parameter n_estimators defines the number of trees in the forest. A higher value generally improves model performance, but also increases computational cost. However, after a certain point, the benefit of adding more trees diminishes, and may eventually plateau. A typical value can be in the range of hundreds to thousands, depending on the size and complexity of the data set.

For all this, no huge values of max_depth and n_estimators were put in the options, in order to not overfit the model.

Best parameters selected with the GridSearch:
- {'max_depth': 50, 'n_estimators': 500}.

Mean and standard deviation of the results of the best model:
- -0.004 and (+/-0.001).

Mean Squared Error on testing dataset:
- 0.004286808596908077.

Mean Squared Error on validation dataset:
- 0.004519706926560613.

On the other hand, the learning curve shows that the testing error decreases faster than the training error, which seems to be stable from the beginning and does not tend to decrease much. Only until the end does it seem that the test error stabilizes, although all the time it was constantly decreasing, this being a sign that the model has learned correctly according to the data provided and did not overfit.

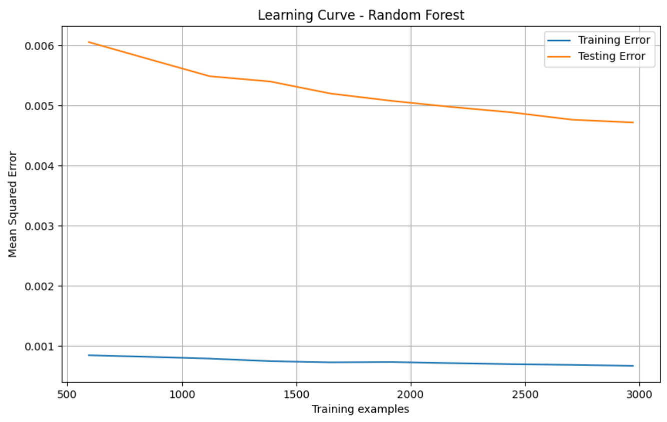

Figure 18: Learning curve – Random Forest.

The result of the most relevant characteristics offered by Random Forest is different from that offered by Multiple Regression. However, Random Forest appears to have a lower error and better performance than Multiple Regression, which is why it is preferable to rely on the classification of features according to their importance that it offers.

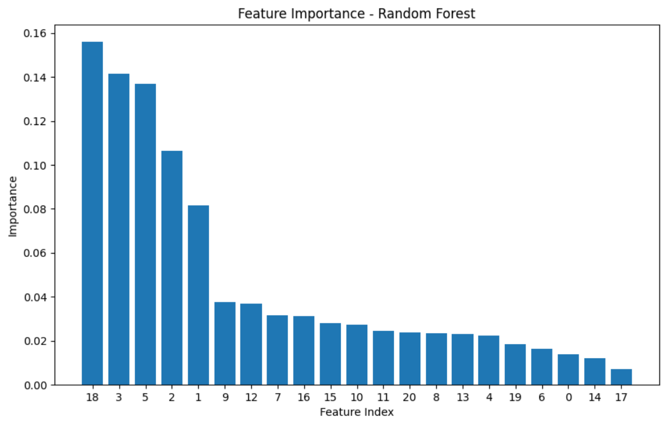

Figure 19: Feature importance – Random Forest.

c) Support Vector Machine:

Parameters for training:

- {'C': [0.1, 1.0, 5.0, 10.0, 100.0], 'kernel': ['linear', 'poly', 'sigmoid', 'rbf'], 'gamma': ['scale', 'auto']}

C is a regularization parameter that controls the trade-off between having a soft decision boundary and correctly classifying the training points. A lower value of C makes the decision boundary softer, allowing for incorrect classifications if that improves generalization. On the other hand, a higher value of C penalizes incorrect classifications more strongly, leading to a tighter decision boundary.

kernel is the parameter that specifies the type of kernel used in the kernel function. Common values are 'linear', 'poly' (polynomial), 'sigmoid' and 'rbf' (radial basis function).

gamma is the parameter that affects the shape of the kernel function and can be understood as the inverse of the kernel width. Lower values mean larger width and therefore larger influence regions for each point.

Best parameters selected with the GridSearch:

- {'C': 5.0, 'gamma': 'scale', 'kernel': 'rbf'}.

Mean and standard deviation of the results of the best model:

- -0.005 and (+/-0.001).

Mean Squared Error on testing dataset:

- 0.005248416336855364 5.

Mean Squared Error on validation dataset:

- 0.005097593802365422.

On the other hand, the learning curve shows that the model had a fast and good learning performance, and although the error at some point wanted to increase, it constantly decreased again until it began to stabilize at the end, being close to starting overfitting.

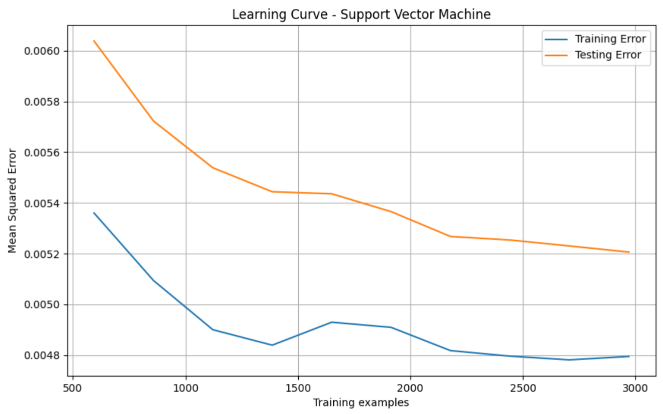

Figure 20: Learning curve – Support Vector Machine.

d) Neural Network:

Parameters for training:

- {'max_iter': [500], 'hidden_layer_sizes': [10, 50, 100, 250, 500], 'activation': ['tanh', 'relu', 'identity', 'logistic'], 'solver': ['adam'], 'alpha': [1e-5, 1e-4, 1e-3, 1e-2, 1e-1], 'tol': [1e-4, 1e-1], 'learning_rate_init': [1e-3, 1e-1], 'learning_rate': ['constant', 'adaptive']}.

The max_iter hyperparameter is the maximum number of iterations, that is, the limit on the number of times the model will adjust the weights during training.

The hidden_layer_sizes hyperparameter is the number and size of the hidden layers of the neural network.

The hyperparameter activation is the activation function used in hidden layers.

The hyperparameter solver is the algorithm used to optimize the weights. 'adam' is a stochastic gradient descent-based optimizer and is the most efficient and used currently. That is why it is the only option.

The hyperparameter alpha is the regularization term to control overfitting.

The hyperparameter tol is the tolerance for convergence. Indicates when to stop training based on loss.

The learning_rate_init hyperparameter corresponds to the initial learning rate, that is, the magnitude by which the weights are adjusted during training.

The learning_rate hyperparameter is the method to adapt the learning rate during training, which can be 'constant' or 'adaptive'. The adaptive learning rate adjusts the rate based on the improvement in validation [8].

Best parameters selected with the GridSearch:

- {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': 250, 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'adam', 'tol': 0.0001}.

Mean and standard deviation of the results of the best model:

- -0.006 and (+/-0.001).

Mean Squared Error on testing dataset:

- 0.006238386845133555.

Mean Squared Error on validation dataset:

- 0.006425751740710489.

On the other hand, the learning curve shows a pattern quite similar to that which occurred with the Support Vector Machine. However, the error does not stabilize at the end and continues to have a fairly steep slope, with a high tendency to continue decreasing. This fortunately means that the model can continue learning and increasing its efficiency if it continues being fed data. However, currently there is no more data, and this is how the neural network, despite nowadays being the basis and structure of deep learning, cannot have an error as minimal as that achieved by Random Forest and SVM, due to the Neural Network requires gigantic amounts of data to achieve its maximum potential.

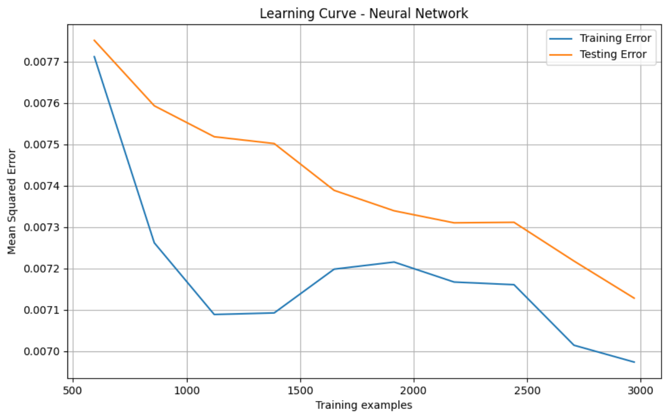

Figure 21: Learning curve – Neural Network.

V. CONCLUSIONS

The results obtained from all machine learning models demonstrate sufficient efficiency and functionality. However, their differences, advantages, and disadvantages are apparent. In general terms, the current preferred choices are Random Forest as the top option, followed by Support Vector Machine. These models exhibited the smallest error both in the training dataset with Cross Validation and when tested with the testing and validation datasets.

Moreover, their learning curves performed well and displayed no indications of overfitting. Although the SVM curve appeared more favorable, the Random Forest is preferred due to its lower Mean Squared Error (MSE) and the additional benefit of providing access to the most relevant features, enabling further analysis.

Table 1: Training results of machine learning models

Model	Mean	Standard deviation	MSE on Testing set	MSE on Validation set	Training duration
Multiple Regression	0.0060932 3	0.018577 74	0.006287 93	0.0064850 2	0.20045 264
Random Forest	0.0041418 7	0.020015 75	0.004286 81	0.0045197 1	42.4779 955
Support Vector Machine	0.0049806 2	0.021461 26	0.005248 42	0.0050975 9	22.8429 364
Neural Network	0.0059789 3	0.020237 69	0.006238 39	0.0064257 5	94.8201 293

On the other hand, if a substantial amount of additional data were to become available, the Neural Network might become the optimal choice. As evident from its learning curve, there is still potential for further improvement and error reduction. However, due to the limited amount of data available, the Neural Network did not achieve its full training potential.

Ultimately, the selection of one model over another depends on the specific circumstances and required conditions. It is crucial to have a clear understanding of the operation of each model, their hyperparameters, and also comprehend other phases of creating machine learning models, such as data preprocessing and information visualization.

VI. REFERENCES

[1] Codigo Maquina. (2021, August 21st). Imputación (o Manejo de Datos Faltantes) con Python. [Video]. YouTube. https://www.youtube.com/watch?v=XiKYdHUsgyM&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=8.

[2] Codigo Maquina. (2021, September 28th). Escalamiento, Normalización y Estandarización de Datos con Python para Ciencia de Datos. [Video]. YouTube. https://www.youtube.com/watch?v=-VuR14Qyl7E&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=7&t=1609s.

[3] Codigo Maquina. (2021, August 8th). Distribución de Datos e Histogramas con Python. [Video]. YouTube. https://www.youtube.com/watch?v=a013caLihMg&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=11.

[4] Codigo Maquina. (2021, August 15th). Diagramas de Caja (BoxPlots) y Datos Anómalos (outliers) con la Prueba de Tukey en Python. [Video]. YouTube. https://www.youtube.com/watch?v=wsfV4AO8UFw&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=12.

[5] Codigo Maquina. (2021, August 2nd). Visualización de Datos usando Matplotlib de Python - Curso introductorio a la creación de gráficas. [Video]. YouTube. https://www.youtube.com/watch?v=foLHkB6W_Ww&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=10.

[6] StatQuest with Josh Starmer. (2019, September 30th). Support Vector Machines Part 1 (of 3): Main Ideas!!!. [Video]. YouTube. https://www.youtube.com/watch?v=efR1C6CvhmE.

[7] Codigo Maquina. (2021, October 12th). Datos de Entrenamiento, Validación y Prueba: ¿Cómo crearlos y qué objetivos tienen? [Video]. YouTube. https://www.youtube.com/watch?v=vdYzm4xC7mc&list=PLUaCK0KaPecOpCSiBi_DH2JBGl_29cvOn&index=16.

[8] AssemblyAI. (2022, February 14th). Neural Networks Summary: All hyperparameters. [Video]. YouTube. https://youtu.be/h291CuASDno?si=DtmRCNrBQe3ndVlk.